May 23, 2025

STOP ANTHROPOMORPHIZING INTERMEDIATE TOKENS AS REASONING/THINKING TRACES!

Subbarao Kambhampati **Kaya Stechly** **Karthik Valmeekam** **Lucas Saldyt** **Siddhant Bhambri**

Vardhan Palod **Atharva Gundawar** **Soumya Rani Samineni** **Durgesh Kalwar** **Upasana Biswas**

School of Computing & AI
Arizona State University

ABSTRACT

Intermediate token generation (ITG), where a model produces output before the solution, has been proposed as a method to improve the performance of language models on reasoning tasks. These intermediate tokens have been called "reasoning traces" or even "thoughts" – implicitly anthropomorphizing the model, implying these tokens resemble steps a human might take when solving a challenging problem. In this paper, we present evidence that this anthropomorphization isn't a harmless metaphor, and instead is quite dangerous – it confuses the nature of these models and how to use them effectively, and leads to questionable research.

1 Introduction

Recent advances in general planning and problem solving have been spearheaded by so-called "Long Chain-of-Thought" models, most notably DeepSeek's R1 [17]. These transformer-based large language models are further post-trained using iterative fine-tuning and reinforcement learning methods. Following the now-standard teacher-forced pre-training, instruction fine-tuning, and preference alignment stages, they undergo additional training on reasoning tasks: at each step, the model is presented with a question; it generates a sequence of intermediate tokens (colloquially or perhaps fancifully called a "Chain of Thought" or "reasoning trace"); and it ends it with a specially delimited answer sequence. After verification of this answer sequence by a formal system, the model's parameters are updated so that it is more likely to output sequences that end in correct answers and less likely to output those that end in incorrect answers with no guarantees of trace correctness.

While (typically) no direct optimization pressure is applied to the intermediate tokens [4, 61], empirically it has been observed that language models perform better on many domains if they output such tokens first [37, 54, 60, 19, 16, 17, 38, 35, 28]. While the fact of the performance increase is well-known, the reasons for it are less clear. Much of the previous work has framed intermediate tokens in wishful anthropomorphic terms, claiming that these models are "thinking" before outputting their answers [37, 12, 17, 55, 61, 7]. The traces are thus seen both as giving insights to the end users about the solution quality, and capturing the model's "thinking effort."

In this paper, we take the position that anthropomorphizing intermediate tokens as reasoning/thinking traces is (1) wishful (2) has little concrete supporting evidence (3) engenders false confidence and(4) may be pushing the community into fruitless research directions. This position is supported by work questioning the interpretation of intermediate tokens as reasoning/thinking traces (Section 4) and by stronger alternate explanations for their effectiveness (Section 6).

Anthropomorphization has long been a contentious issue in AI research [32], and LLMs have certainly increased our anthropomorphization tendencies [20]. While some forms of anthropomorphization can be treated rather indulgently as harmless and metaphorical, our view is that viewing ITG as reasoning/thinking is more serious and may give a false sense of model capability and correctness.

The rest of the paper is organized as follows: We will start in Section 2 by giving some background on the main ideas behind reasoning models, with special attention to post-training on derivational traces.[1] In Section 3, we will discuss the evidence for and ramifications of anthropomorphizing intermediate tokens as reasoning traces. In Section 4, we directly consider the question of whether intermediate tokens can be said to have any formal or human-interpretable semantics. In Section 5, we look at the pitfalls of viewing intermediate tokens as computation that is adaptive to problem complexity. Section 6 looks at some potential ways of making sense of the performance of LRMs that don't depend on anthropomorphizing intermediate tokens. We will end in Section 7 with a summary of our position and the downsides of anthropomorphizing intermediate tokens.

Before going forward, we should clarify some potential confusion regarding the "reasoning trace" terminology. By intermediate tokens, we refer to the unfiltered tokens emitted by the LLM before the solution. This should be distinguished from post-facto explanations or rationalizations of the process or the product of said "thinking." For example, OpenAI o1 *hides* the intermediate tokens it produces (perhaps because they aren't that interpretable to begin with?) but sometimes provides a sanitized summary/rationalization instead. In contrast, DeepSeek R1 [9] provides the full intermediate token sequences (which often run for pages even for simple problems). To be clear, our focus here is on the anthropomorphization of unfiltered intermediate tokens rather than such post-facto rationalizations. It is well known that for humans at least, such post-facto exercises are meant to teach/convince the listener, and may not shed much meaningful light on the thinking that went in [36].

2 Background: Test Time Inference & Post-Training in Reasoning Models

Large Language Models (LLMs), which have been autoregressively trained on humanity's digital footprint, have shown the ability to generate coherent text responses to a vast variety of prompts. Although they show impressive System 1 capabilities, and excel in producing completions that mimic style, System 2 capabilities like factuality, reasoning, and planning have remained elusive aims, if not Achilles heels [21].

In response, researchers have developed a new breed of models – sometimes called *Large Reasoning Models* (LRMs) – which build on vanilla LLM architectures and training recipes. The best-known of these are OpenAI's o-series of models o1, o3, DeepSeek's R1 [9], Google Gemini-2.5-pro, Anthropic Claude 3.7 Sonnet, which have shown significant performance improvements on reasoning and planning tasks previously outside the range of older LLM capabilities.

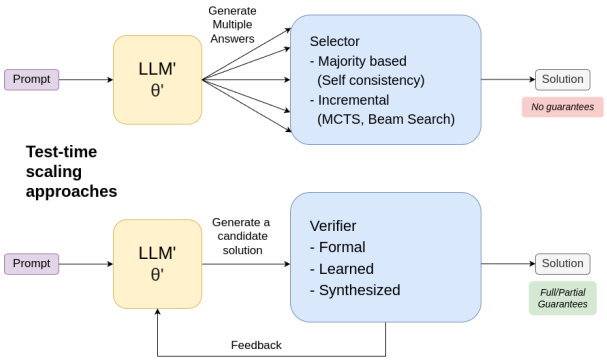

Figure 1: Test-time scaling approaches for teasing out reasoning

These models have been built on insights from two broad but largely orthogonal classes of ideas:
(i) **test-time inference** scaling techniques, which involve getting LLMs to do more work than simply providing the most likely direct answer; and (ii) **post-training methods**, which complement simple auto-regressive training on web corpora, with additional training on intermediate token data.

2.1 Test-time Inference

Not all problems require an equal amount of effort or time. A two digit by two digit addition problem can be solved with just three one-digit additions, while a four by four digit problem may require seven. There is a rich history of approaches that use scalable online computation to improve upon faster initial guesses, including limited depth min-max, real-time A* search and dynamic programming, and Monte Carlo Tree Search[42, 15]. Test-time inference approaches (see Figure 1) mirror these ideas.

Perhaps the most popular and enduring class of test-time inference ideas involves generating many candidate solutions from an LLM and using some selection procedure to choose the final output. The simplest implementation is known as *self-consistency*[53]: choose the most common answer. Total time spent is proportional to the number of solutions generated, but while this method can work practically, it provides no guarantees that its answers will be more correct.

[1]We will use the term *derivational trace* as a neutral stand-in for intermediate tokens, whether generated by humans, formal solvers or other systems, rather than the more popular anthropomorphized phrases "Chains of thought" and "reasoning traces".

More sophisticated selection procedures attempt to verify that an LLM's output is correct. When paired with an LLM in this manner, the combined system can be seen as a *generate-test* framework, and naturally raises questions about the verification process: *who does it*, and *with what guarantees?* A variety of approaches have been tried–including using LLMs themselves as verifiers[56] (although this is known to be problematic [48]), learning verifiers[2, 58], and using external sound verifiers that come with either full or partial guarantees. In cases where verifiers provide explanations or feedback when a guess is incorrect, these can be passed back to the LLM so it generates better subsequent guesses. Several well-known LLM-based reasoning systems such as FunSearch [41], Alpha Geometry [51] and AlphaEvolve [1] all can be viewed under this lens. The LLM-Modulo framework[22, 21] provides an umbrella for these types of verification-based approaches, along with their guarantees, which are essential when these systems are deployed in safety-critical applications, or even in conventional applications where wrong answers are unacceptable.

2.2 Post-Training on Derivational Traces

Unlike the test-time inference techniques, that augment the inference stage of standard LLMs, the post-training training techniques are aimed at the LLM training stage.[2] Standard LLMs are trained using a very simple objective: given a chunk of text, predict the most likely next token. This procedure, when employed with sufficiently high capacity models on web-scale corpora, has been surprisingly successful at capturing diverse text styles. The sheer variety of linguistic training data they've ingested opens up the possibility of applying them to nearly any domain, including reasoning and planning. However, while sufficiently accurate mimicry on peta-scale corpora might be enough to hypothetically succeed at these tasks, vanilla LLMs struggle at planning and reasoning. Their completions almost always look reasonable despite often being incorrect[21], seemingly relying on statistical features and stylistic quirks rather than robust procedures.

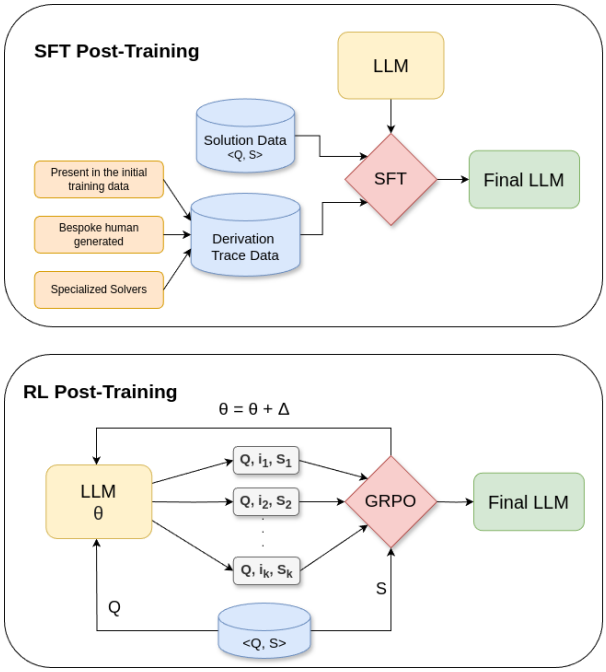

Figure 2: Post-training Approaches for teasing out reasoning

One intuition driving today's research is that this performance gap is partly because the training data is incomplete. LLMs have soaked up every article, post, and book on the internet but not what it took to produce them – whether internal verbalizations, scratch paper outlines, or typed up but discarded drafts. Perhaps, the hope here goes, if more of these *derivational traces* were included, this would help LLMs replay versions of the same processes.

While promising, it is far from immediately clear how to source data like this at sufficient scale. There are few if any large collections of generic derivational traces. Not only is it burdensome for people to produce granular step-by-step representations of their own thoughts, but they are unlikely to have direct and explicit access to those processes in the first place. And in those cases where they do, they may deliberately or subconsciously efface their tracks. As Gauss famously remarked when asked to give step-wise intuitions for his proofs: no self-respecting architect leaves the support structure in place once the edifice is complete!

Nevertheless, a variety of approaches have tried to make up for this shortfall, ranging from paying annotators for step-by-step derivations to generating and selecting them with LLMs. We classify these in terms of (i) how candidate traces are generated and filtered, and (ii) how they are used to improve the underlying LLM through supervised fine tuning or reinforcement learning; see Figure 2.

Before diving into the details, we should point out that the gap between this anthropomorphic motivation in terms of internal thoughts and actual LLM operation is quite stark. Often, the "derivational traces" used in practice don't have any systematic relation to robust reasoning processes, despite resulting in empirical performance improvements. We elaborate this point in subsequent sections.

[2] As we argue in Section 6.1, can be seen as compiling test-time verification into the model at the training time.

Generating Candidate Derivational Traces: Several trace generation methods were considered:

Human-generated Traces: An obvious way to obtain additional derivational data is to have humans create it. OpenAI paid contractors to write questions and step by step solutions to grade school math problems to create GSM8k[29]. While companies have continued to source data like this, it is infeasibly expensive, especially at the data scales necessary for large scale model training and for the diversity of problems that require supporting derivational data.

Solver-generated Traces: A much more scalable approach is to use formal solvers to automatically generate both solutions and rationales derived from solver-specific intermediate representations. Searchformer[26], Stream of Search[13], as well as DeepMind's work in [44, 31] use standard search algorithms to produce datasets containing not just answers but also the execution traces generated along the way. For instance, when using A* search to solve a problem, SearchFormer's data generation pipeline will provide a representation of each manipulation of the open and closed lists as a derivational trace. Unfortunately, domain-specific solvers cannot be used to generate traces for arbitrary problems, limiting the generality of this technique.

LLM-generated Traces: Rather than creating high-quality traces from the start, an increasingly popular approach is to generate them from an LLM and filter afterwards. This sort of generation is feasible because modern LLMs are pre-trained on data that already contains some derivational traces (e.g. educational web pages, grade school math explanations, and other sources with steps)[3], and outputs that match these styles can be reliably induced, often by merely appending "Let's think step by step" to the prompt and hoping for traces that might loosely resemble reasoning [23].

Filtering Traces: Naively LLM-generated traces are often not useful unless they are filtered. Researchers have varied in how they approach this trace selection process, ranging from selecting only those that are correct at each step (according to human labelers), training process reward models that attempt to automate human verification[29], to selecting traces by formally verifying whether they lead to correct final solutions without considering the trace content [57, 9].

Improving LLMs Using Derivational Traces: Once derivational traces have been selected, they can be used to further train an LLM. The hope is that, by outputting useful intermediate tokens, the LLM will be more likely to output correct solutions across a wider variety of problems. Early approaches fine-tuned LLMs directly on such traces[57, 26, 13], but more recent advances have pivoted towards using reinforcement learning (RL) instead (although there are questions about the generality of the MDP models used in the current LLMs like DeepSeek R1; see Section 5).

The first major successful and publicly understood models trained this way were DeepSeek's R1-Zero and R1 models[9]. After completing normal LLM pre-training, they begin an RL post-training phase on a new dataset – consisting of questions whose answers can be automatically verified. During this phase, the LLM generates multiple possible completions for each question; these completions take the form of traces culminating in separately marked final answers, and are scored according to the correctness of that final answer. The best completions are then rewarded, adjusting the model parameters to be more likely to output them rather than those completions that did not lead to a correct final answer. In essence, this RL process views the LLM as a token-choosing policy and uses a policy gradient algorithm to iteratively improve its parameters. The "state" here is the context window; the next action is just the token emitted by the policy (see Section 5).

Conceptually, this RL phase can be considered a two step process repeated many times: first, generate potential trajectories from the LLM and weight them using an automatically computed success criterion; second, selectively fine-tune the same LLM on its own output. Whether SFT or RL is used to modify the parameters of the base LLM, the resulting model's architecture is still the same as that of any other LLM. The only difference is in the probability distribution the model captures: one that favors outputting intermediate tokens (which mimic the derivational traces it was trained on) followed by the LLM's guess at the solution. This reframing makes it clear that pure fine-tuning and RL approaches are not as different as might be initially assumed, supported by [43].

3 Consequences of Anthropomorphizing Intermediate Tokens

As we discussed, post-training can induce a model to first generate long strings of intermediate tokens before outputting its final answer. There has been a tendency in the field to view these intermediate tokens as the human-like "thoughts" of the model or to see them as *reasoning traces* which could reflect internal reasoning procedures. This is precisely

[3]There is also some speculation that the popularity of chain of thought prompting techniques has led to a greater availability of diverse step by step trace data in the massive web crawls that make up much of pre-training data.

the tendency our position paper argues against. We start by listing the various (unhealthy) ramifications of this anthropomorphization:

- Viewing intermediate tokens as reasoning/thinking traces has led to a drive to make them "interpretable" to humans in the loop (nevermind that interpretability mostly meant that the traces were in pseudo English). For example, DeepSeek [9] dabbled in training an RL-only model (R1-Zero) but released a final version (R1) that was trained with additional data and filtering steps specifically to reduce the model's default tendencies to produce intermediate token sequences that mix English and Chinese!
- It has led to an implicit assumption that correctness/interpretability of the intermediate tokens has a strong correlation, or even causal connection, with the solution produced. This tendency is so pronounced that a major vendor's study showing that LRM's answers *are not always faithful* to their intermediate tokens was greeted with surprise [8].
- Viewing intermediate tokens as traces of thinking/reasoning has naturally led to interpreting the *length* of the intermediate tokens as some sort of meaningful measure of problem [49, 50] difficulty/effort and techniques that increased the length of intermediate tokens were celebrated as "learning to reason" [9]. Simultaneously there were efforts to *shorten* intermediate traces produced and celebrate that as learning to reason efficiently [3].
- There have been attempts to cast intermediate tokens as learning some "algorithm" that generated the training data. For example, the authors of SearchFormer [26] claim that their transformer learns to become "more optimal" than A* because it produces shorter intermediate token traces than A*'s derivational trace on the same problem.

These corollaries, in turn, have lead to research efforts, which, when viewed under the lens of our position, become questionable enterprises (as we shall discuss in the following sections).

4 On the Amorphous Semantics of Intermediate Tokens

The fact that intermediate token sequences often reasonably look like better-formatted and spelled human scratch work – mumbling everything from *"hmm..."*, *"aha!"*, *"wait a minute"* to "interesting." along the way – doesn't tell us much about whether they are used for anywhere near the same purposes that humans use them for, let alone about whether they can be used as an interpretable window into what the LLM is "thinking," or as a reliable justification of the final answer.

Famously, DeepSeek's R1 paper claimed that one of the most impressive observed behaviors of their trained models was the so-called "aha" moment: as part of the chain of thought it was producing in order to answer some question, the model output the token "aha", seeming to indicate that it had come upon a sudden realization. While a human may say "aha" to indicate exactly a sudden internal state change, this interpretation is unwarranted for models which do not have any such internal state, and which on the next forward pass will only differ from the pre-aha pass by the inclusion of that single token in their context. Interpreting the "aha" moment as meaningful exemplifies the long-neglected assumption about long CoT models – the false idea that derivational traces are semantically meaningful, either in resemblance to algorithm traces or to human reasoning. Further, there have also been works which attribute cognitive behaviors (like backtracking, self-verification etc.) to the models based on their reasoning traces and try to induce these kinds of behaviors through examples in the hope of improving the models' performance [12, 40].

One reason that this anthropomorphization continues unabated is because it is hard to either prove or disprove the correctness of these generated traces. DeepSeek's R1, even on very small and simple problems, will babble over 30 pages worth of text in response to each and every query, and it is far from clear how to check if these monologues constitute sound reasoning.[4] While there have been some valiant efforts to make sense of these large-scale mumblings– e.g. [30]–the analyses here tend to be somewhat qualitative and suggestible reminiscent of "lines of code" analyses in software engineering. It is no wonder then that few if any LRM evaluations even try to check their pre-answer traces, and focus only on evaluating the correctness of their final answers.[5]

However, while evaluating the intermediate tokens produced by general LRMs may be out of direct reach, we *can* formally verify the status of traces generated by format-constrained models trained to imitate the derivational traces of domain-specific solvers. In [45] the authors challenge the prevailing narrative that intermediate tokens or "Chains of

[4]Before DeepSeek, the entire question was moot. OpenAI's o1 model deliberately hides its intermediate tokens from end users, despite charging based on how many were produced!

[5]Approaches like Process Reward Models [59] try to make the reasoning traces a bit more locally consistent–but they seem to have taken a back seat since the success of DeepSeek R1.

Thought" generated by Large Reasoning Models like DeepSeek's R1 are interpretable, semantically valid sequences with predictable effects on the model's behavior. As they didn't have access to any frontier LLM's training data or even exact training procedure, and since the traces these models output are in multiply-interpretable natural language without a concrete ground truth, they design a series of experiments building on previous smaller model reasoning work – mainly Searchformer and Stream of Search [14, 27] – and construct an A* trace validator, finding that there is only a loose correlation between the correctness of the trace and the correctness of the output plan. They then report a causal intervention, training additional models on noisy or irrelevant traces and find that there are (nonsensical) trace formats that nevertheless maintain or even increase the model's performance – all despite them being much less informative or connected to the problem at hand.

Presumably, natural language reasoning follows algorithmic structure, even if it does not correspond to a rigidly-defined algorithm. For example, see Polya's "How to Solve It," [39] which outlines the elements of mathematical problem solving in an algorithmic way, even if they are often implicit. Accordingly, we argue that studying algorithmic search traces, such as in [46], resembles a *model organism* for understanding systems like R1 (analogous to the roles of *Drosophila Melanogaster* or *Caenorhabditis Elegans* in biology). If a technique can learn to produce semantic reasoning traces for natural language problems, it ought to be able to do so for algorithmic traces as well, and vice-versa. Accordingly, evidence that models trained on algorithmic traces do not learn semantics applies to natural language problems and systems that apply to them, namely R1.

A similar investigation to test the correlation, and potentially any causation, between intermediate traces and final solution performance was carried out by the authors in [5] in the Question-Answering (QA) domains. By decomposing the QA reasoning problems into verifiable sub-problems that can be evaluated at inference time, the authors first generated a Supervised Fine-Tuning (SFT) dataset with correct intermediate traces paired with correct final solutions. To carry out an intervention experiment, they generate another SFT dataset consisting of incorrect intermediate traces again paired with correct final solutions. For the first SFT experiment setting, the results show a large number of False Positives where the fine-tuned models output correct final solutions but incorrect intermediate traces. Interestingly, the intervention experiments with incorrect intermediate traces even outperforms the SFT with correct intermediate trace setting.

Li et al.[28] perform model distillation using noisy traces on math and coding problems and find that the smaller LLM that is being trained remains largely robust to the semantic noise in the trace. Even when trained on derivational trace containing largely incorrect mathematical operation, the LLM shows significant performance improvements as compared to the base model. Dualformer [49], an extension of Searchformer [27], which trains transformer models on truncated A* derivational traces (by arbitrarily removing steps from the original A* search process–and thus destroying any trace semantics) to improve solution accuracy, is another evidence for performance improvements with wrong traces!

If the intermediate tokens produced by models that are explicitly trained on correct traces are still not guaranteed to be valid during inference time, then there seems to be little reason to believe that trace validity improves when these models are further post-trained with RL or incremental SFT. This is because such post-training techniques [9, 43] change the base model parameters to bias it more towards the trajectories that end up on solutions verified correct by the external verifiers during training. Most works that do these types of post-training reward only the solution accuracy and ignore the content of intermediate tokens [9].

Given that these traces may not have any semantic import, deliberately making them *appear* more human-like is dangerous. A In the end, LRMs are supposed to provide solutions that users don't already know (and which they may not even be capable of directly verifying). Engendering false confidence and trust by generating stylistically plausible ersatz reasoning traces seems ill-advised! After all, the last thing we want to do is to design powerful AI systems that potentially exploit the cognitive flaws of users to convince them of the validity of incorrect answers.

token

5 Intermediate Token Production and Problem Adaptive Computation

Although our main focus is on the anthropomorphization and semantics of derivational traces, a related aspect is the extent to which traces reflect learned procedures or problem adaptive computation. When an LRM is generating *more intermediate tokens* before providing the solution, it is clearly doing more computation, but the nature of this computation is questionable, as is interpreting it as a meaningful procedure. The question is whether this computation reflects an intended procedure, and then if the length of computation can be viewed meaningfully as adaptive to problem difficulty.

Interestingly, there has been a tendency to celebrate post-training techniques for *increasing* the intermediate token length. DeepSeek R1 [9], for example, claims that RL post-training is *learning to reason* as shown by the increased length of intermediate tokens over RL epochs. It is even more ironic that there have been subsequent efforts to *reign in* the intermediate token lengths, and claim that as a way to reduce compute while preserving task performance/accuracy (c.f. [3]).

Part of this misconception comes from the simplistic MDP formulation adopted by DeepSeek R1 and subsequent work [17]. In [43, 10] the authors examine this formulation, showing that with the structural assumption of representing states as sequences of tokens, and uniformly distributing the terminal reward into intermediate tokens, RL is incentivized to generate longer intermediate token sequences–something that has been misattributed to "improved reasoning." At some level, this shouldn't be surprising given that the whole point of RL is to figure out credit assignment, and the division of final reward equally into intermediate tokens short circuits this process in an *ad hoc* way.

Given that the increased length of intermediate tokens is celebrated by DeepSeek R1 [9], the fact that these may be happening due to a rather simplistic way of equally dividing advantage over all tokens should temper the credibility of claims that longer intermediate tokens in systems like R1 [9] are automatically indicative of "thinking effort."

6 Understanding LRMs without Anthropomorphizing Intermediate Tokens

Anthropomorphization of the intermediate tokens as reasoning/thinking traces has provided a comforting explanation of the observed performance of LRMs. Our arguments in this paper foreground the possibility that this is a *cargo cult explanation* [11], namely that derivation traces resemble reasoning *in syntax only*. This leads to the question as to what are other plausible explanations of the effectiveness of LRMs? While the main focus of this position paper is to caution the community away from questionable explanations, rather than to provide complete explanations of the source of the power of LRMs we do present some plausible candidate explanations below:

6.1 Reasoning as Incremental Learning of Verifier Signal

Most documented advances of LRMs on reasoning problems have been on tasks for which there are formal verifiers from traditional AI and Computer Science. The *modus operandi* of current LRMs is leveraging these verifiers in a *generate-test* loop at test time, training time or distillation time in order to partially compile/internalize the verification signal into generation. In other words, post-training LRMs can be seen as iteratively compiling reasoning into retrieval via learning.

Internalizing reasoning is needed because, for reasoning problems which can be arbitrarily scaled in complexity (e.g. multi-digit multiplication with increasing digit numbers), an LLM trained on instances of a certain size quickly loses its ability to provide good guesses at larger sizes[47]. As we have seen, post-training approaches depend on the ability of the base LLM to have high enough top-k accuracy (i.e., be capable of generating at least one correct solution given k guesses) so that the verifier has something to select (otherwise, there is no signal either for fine tuning or the RL phase!).

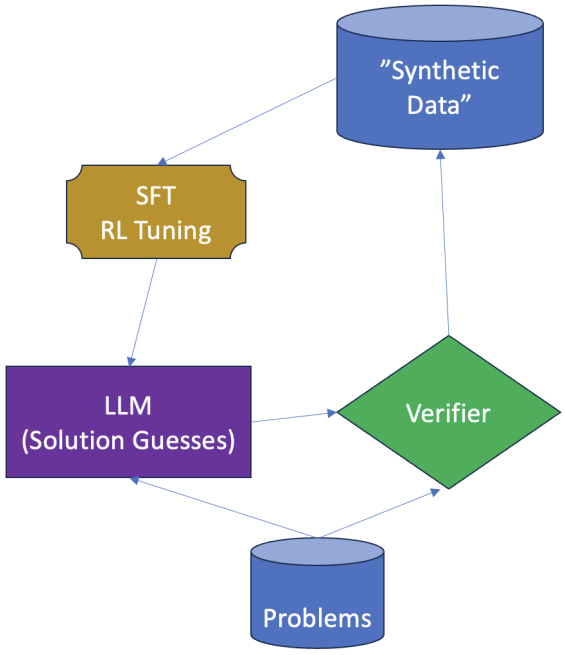

Figure 3: Understanding LRM Improvement as Incremental Compilation of Verifier Signal

This general idea mirrors Marvin Minsky's insight that *intelligence is shifting the test part of generate-test into generation [34].* In particular, using verifiers at test time has already been advocated by the LLM-Modulo framework[22]. As we saw in Section 2.2, LRM post-training approaches crucially depend on the signal from the verifier to separate trajectories supplied by the base LLM into those that reach correct solutions vs. those that don't (and thus, this can be seen as a form of "train time LLM-Modulo"). Once this is done, these traces are used to refine the base LLM ("generator") via either finetuning or RL. This refinement can thus be interpreted as incrementally

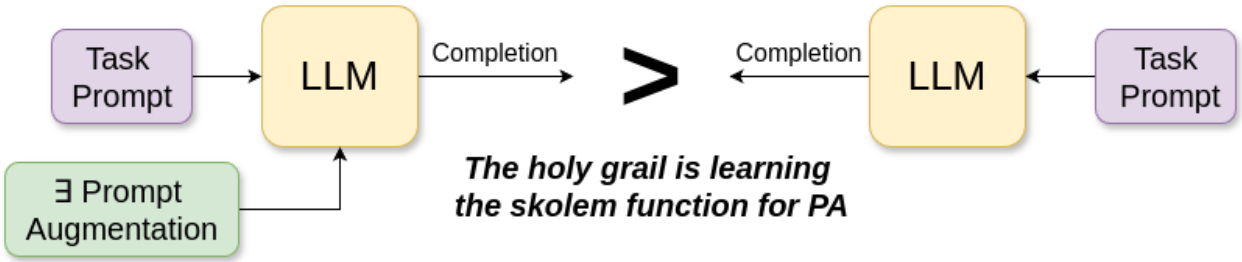

Figure 4: Augmenting a task prompt with additional tokens often seems to improve the accuracy of LLM completion even if the tokens don't have human-parseable meaning.

compiling the verifier signal into the generator. Finally, while Deepseek R1 just deploys the refined LLM at inference stage, without resorting to any test time verification, they do wind up using verifiers when they develop additional synthetic data with the help of R1 to distill other models.

One way of seeing this training-, test-, and distillation-time verification is as a staged approach to compile the verification signal into an underlying LLM (see Figure 3). In particular, as we discussed, the base LLM used for R1 already has the capability of generating plausible solution trajectories (potentially from the derivational trace data that was already present in the pre-training data). Post-training can be seen as further refining it to come up with accurate solutions for longer/harder problems in fewer tries. Distillation can be seen as propagating this even further. At each stage, the verification signal is being internalized into the underlying LLM for longer and longer "inference horizons." This understanding is consistent with studies on the effectiveness of Chain of Thought[47], use of internal vs. external planning approaches for games[44], as well as self-improvement in transformers[25]. In the last case, we would qualify any "self-improvement" claims by saying that it is more the case of incrementally compiling the verifier signal.

6.2 Embracing Reason-less Intermediate Tokens

One reasonable question about our position is *So what if the intermediate traces don't have semantics? We can just hide them from end user (like OpenAI o1/o3 do)* . We believe that a half-hearted lip service to human-legibility properties can not only engender false trust in the solutions (as already discussed), but also can become an albatross if our goal is increase task performance. This is already hinted by experiments in works such as [46, 6] that show that performance can improve when the model is trained on incorrect traces! Even the DeepSeek R1 authors [9] admit that R1-Zero, which mixed English and Chinese intermediate tokens, actually had better performance than the subsequent R1 (that starts with a base model further SFT'd on thousands of human-annotated reasoning traces!).

Reinforcement learning can potentially train LLMs to output any old intermediate token sequences – all that matters is that the bottom line improves. Indeed, we believe that de-anthropomorphization of intermediate tokens starts by acknowledging the common assumption across most "chain of thought" approaches: that an LLM will generate more accurate completions when provided with an appropriate *prompt augmentation* rather than just the base task prompt (see Figure 4). The big question then is how to get the right prompt augmentation.

That is, given a task prompt T,

$$\exists PA s.t. Pr(Sol(LLM(T + PA), T)) > Pr(Sol(LLM(T), T)),$$

where PA is some appropriate prompt augmentation, $LLM(x)$ is the completion output by LLM given x as the prompt, and $Sol(y, T)$ checks, with the aid of a verifier, if y contains a solution for T.

The holy grail then is learning the Skolem function that supplies the right prompt augmentation that increases the probability of producing the correct answer in the succeeding tokens. The fact that we have an existential in the prompt augmentation inequality above means that in the most general case, the PA may be a function of both the task and the model. Note that *there is nothing here saying that PA must make any sense to the humans or be a correct trace of some algorithm.*

Zero-shot [24] and k-shot chain of thought prompting [54, 47], as well as the variety of approaches for getting derivational traces for post-training that we discussed in Section 2.2, can all be seen as various *heuristic ways* of supplying this prompt augmentation function. (Indeed, we can understand work on LLM adversarial attacks[62, 33], and the work on using tokens from continuous latent space [18] from this perspective!).

It is worth investigating approaches that aim to learn the Skolem function supplying prompt augmentations more directly. One idea is to use a set up where prompt augmentations are proposed by a separate second "actor" LLM (c.f. [52]),

which are then applied to the base "environment" LLM. The goal would be to learn an intermediate token application policy for the actor LLM focused only on improving solution accuracy (see [52]) of the base LLM. This set up can be formalized as reinforcement learning in a general MDP framework (without the need for the simplistic structural assumptions discussed in Section 5), and approaches such as those used in AlphaZero and MuZero can be employed.

We end this section by reiterating that our aim here is to show that there are alternate ways of understanding intermediate tokens as prompt augmentations that don't require anthropomorphization.

7 Summary

In this position paper, we argued against the prevalent tendency to anthropomorphize intermediate tokens as reasoning or "thinking". Anthropomorphization has been a part of AI research [32], and has significantly increased in the era of LLMs [20]. While some anthropomorphization has been harmless metaphors, we argued that viewing intermediate tokens as reasoning traces or "thinking" is actively harmful, because it engenders false trust and capability in these systems, and prevents researchers from understanding or improving how they actually work. We collated emerging evidence to support our position, and offered some more supported and balanced alternate ways of viewing LRM performance and the role of intermediate tokens. Our hope is that this position catalyzes the community towards more fruitful research directions to understand frontier models.

8 Acknowledgements

This research is supported in part by ONR grant N0001423-1-2409, DARPA grant HR00112520016, and gifts from Qualcomm, J.P. Morgan and Amazon.

References

[1] AlphaEvolve: a coding agent for scientific and algorithmic discovery, 2025.

[2] Daman Arora and Subbarao Kambhampati. Learning and leveraging verifiers to improve planning capabilities of pre-trained language models. *ICML Workshop on Knowledge and Logical Reasoning in the Era of Data-driven Learning*, 2023.

[3] Daman Arora and Andrea Zanette. Training language models to reason efficiently, 2025. *URL https://arxiv.org/abs/2502.04463*.

[4] Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 2025.

[5] Siddhant Bhambri, Upasana Biswas, and Subbarao Kambhampati. Interpretable traces, unexpected outcomes: Investigating the disconnect in trace-based knowledge distillation, 2025.

[6] Siddhant Bhambri, Upasana Biswas, and Subbarao Kambhampati. Interpretable traces, unexpected outcomes: Investigating the disconnect in trace-based knowledge distillation, 2025.

[7] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[8] Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner Fabien Roger Vlad Mikulik, Sam Bowman, Jan Leike Jared Kaplan, et al. Reasoning models don't always say what they think.

[9] DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025.

[10] Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

[11] Richard P Feynman. Cargo cult science, 1974.

[12] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.

[13] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of Search (SoS): Learning to Search in Language. In *Conference on Language Modeling (COLM)*, 2024.

[14] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

[15] Alex Graves and Google Deepmind. Adaptive Computation Time for Recurrent Neural Networks. 0 0.

[16] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

[17] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[18] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2024.

[19] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

[20] Lujain Ibrahim and Myra Cheng. Thinking beyond the anthropomorphic paradigm benefits llm research, 2025.

[21] Subbarao Kambhampati. Can large language models reason and plan? *Annals of the New York Academy of Sciences*, 1534(1):15–18, 2024.

[22] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024.

[23] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[24] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[25] Nayoung Lee, Ziyang Cai, Avi Schwarzschild, Kangwook Lee, and Dimitris Papailiopoulos. Self-improving transformers overcome easy-to-hard and length generalization challenges, 2025.

[26] Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond A*: Better Planning with Transformers via Search Dynamics Bootstrapping. In *Conference on Language Models (COLM)*, 2024.

[27] Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping. *arXiv preprint arXiv:2402.14083*, 2024.

[28] Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G Patil, Matei Zaharia, et al. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *arXiv preprint arXiv:2502.07374*, 2025.

[29] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

[30] Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, Nicholas Meade, Dongchan Shin, Amirhossein Kazemnejad, Gaurav Kamath, Marius Mosbach, Karolina Stańczak, and Siva Reddy. Deepseek-r1 thoughtology: Let's think about llm reasoning, 2025.

[31] Larisa Markeeva, Sean Mcleish, Borja Ibarz, Wilfried Bounsi, Olga Kozlova, Alex Vitvitskyi, Charles Blundell, Tom Goldstein, Avi Schwarzschild, and Petar Veličkovi´veličkovi´c. The CLRS-Text Algorithmic Reasoning Language Benchmark. Technical report, 2024.

[32] Drew McDermott. Artificial intelligence meets natural stupidity. *SIGART Newsl.*, 57:4–9, 1976.

[33] Rimon Melamed, Lucas H. McCabe, Tanay Wakhare, Yejin Kim, H. Howie Huang, and Enric Boix-Adsera. Prompts have evil twins. In *Proc. EMNLP*, 2024.

[34] Marvin Minsky. *Society of mind*. Simon and Schuster, 1986.

[35] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[36] Richard E Nisbett and Timothy D Wilson. Telling more than we can know: Verbal reports on mental processes. *Psychological review*, 84(3):231, 1977.

[37] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. 2021.

[38] Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint arXiv:2404.15758*, 2024.

[39] George Polya. How to solve it: A new aspect of mathematical method. In *How to solve it*. Princeton university press, 2014.

[40] Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. To backtrack or not to backtrack: When sequential search limits model reasoning. *arXiv preprint arXiv:2504.07052*, 2025.

[41] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, pages 1–3, 2023.

[42] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. London, 2010.

[43] Soumya Rani Samineni, Durgesh Kalwar, Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Rl in name only? analyzing the structural assumptions in rl post-training for llms, 2025.

[44] John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada Lewis, Anian Ruoss, Tom Zahavy, Petar Veličković, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomašev. Mastering board games by external and internal planning with language models, 2024.

[45] Kaya Stechly, Karthik Valmeekam, Atharva Gundawar, Vardhan Palod, and Subbarao Kambhampati. Beyond semantics: The unreasonable effectiveness of reasonless intermediate tokens, 2025.

[46] Kaya Stechly, Karthik Valmeekam, Atharva Gundawar, Vardhan Palod, and Subbarao Kambhampati. Beyond semantics: The unreasonable effectiveness of reasonless intermediate tokens, 2025.

[47] Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. Chain of Thoughtlessness: An Analysis of CoT in Planning. In *Proc. NeurIPS*, 2024.

[48] Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the Self-Verification Limitations of Large Language Models on Reasoning and Planning Tasks. In *Proc. ICLR*, 2025.

[49] DiJia Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuandong Tian, and Qinqing Zheng. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces. In *The Thirteenth International Conference on Learning Representations*, 2024.

[50] Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms, 2025.

[51] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[52] Karthik Valmeekam, Kaya Stechly, Atharva Gundawar, and Subbarao Kambhampati. A systematic evaluation of the planning and scheduling abilities of the reasoning model o1. *Transactions on Machine Learning Research*.

[53] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[54] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[55] Shu Yang, Junchao Wu, Xin Chen, Yunze Xiao, Xinyi Yang, Derek F Wong, and Di Wang. Understanding aha moments: from external observations to internal mechanisms. *arXiv preprint arXiv:2504.02956*, 2025.

[56] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[57] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

[58] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction, 2024.

[59] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning, 2025.

[60] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

[61] Hengguang Zhou, Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. R1-zero's" aha moment" in visual reasoning on a 2b non-sft model. *arXiv preprint arXiv:2503.05132*, 2025.

[62] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.